Rule WLM200: Average CPU use per transaction is higher than goal

Finding: CPExpert has determined that the average CPU time per transaction was higher than the response goal for the service class. This finding does not apply to subsystem transactions (that is, it does not apply to CICS or IMS transactions).

Impact: This finding has a HIGH IMPACT on performance of your computer system.

Logic flow: The following rules cause this rule to be invoked:

Rule WLM101: Service Class did not achieve average response goal
Rule WLM102: Service Class did not achieve percentile response goal

Discussion: Transactions executing in the system can be in a variety of states from the perspective of the Workload Manager: using the CPU, delayed for an identifiable reason, or delayed for some unknown reason. The System Resources Manager (SRM) periodically samples the state of each address space in each service class. These samples are accumulated into variables which are recorded by RMF in the "Service Class Period Data Section" of SMF Type 72 (Subtype 3) records. Please see Section 4 for a discussion of these states and the sampling process.

CPExpert analyzes the amount of CPU time used by transactions by the following process:

- CPExpert first computes the number of samples which found an address space executing in the service class. This is done by summing CPU Using samples (R723CCUS), Total Wait samples (R723CTOT), and Unknown Delay samples (R723CUNK). The result is titled "EXSAMP" in the code.

- CPExpert divides the number of CPU Using samples (R723CCUS) by the EXSAMP value, to yield the percent of execution samples in which the SRM found an address space was using the CPU. The resulting percentage is multiplied by the average transaction response time to yield the amount of time when the average transaction was using the CPU.

CPExpert compares the amount of time when the average transaction was using the CPU against the response goal. CPExpert produces Rule WLM200 if the CPU use per transaction is higher than the response goal.

The following example illustrates the output from Rule WLM200:

```
RULE WLM200: AVERAGE CPU USE PER TRANSACTION IS HIGHER THAN GOAL

   The average CPU time was higher than the response goal for Service
   Class ST_USERS (Period 1).  The average transaction used more CPU
   time than the response goal of 0.200.  MVS cannot achieve the
   response goal unless the CPU requirements of the average transaction
   can be reduced.  Alternatively, you can review the response goal to
   see whether the goal should be increased.  Please review the discussion
   with WLM200 regarding other alternatives.  This situation applies to the
   following measurement intervals:

                                      TOTAL           AVERAGE CPU TIME
   MEASUREMENT INTERVAL            TRANSACTIONS       PER  TRANSACTION
   14:00-14:15,01MAR1994                14                 0.493
   14:15-14:30,01MAR1994                33                 1.770
   14:30-14:45,01MAR1994                33                 2.553
   14:45-15:00,01MAR1994               198                 0.556
   15:00-15:16,01MAR1994                33                 2.391
```

Suggestion: MVS cannot achieve the specified response goal for the service class unless the CPU requirements of the average transaction can be reduced.

CPExpert suggests that you consider the following actions:

- Perform a "reality" check on the finding from CPExpert by examining the "Response Time Distribution" produced by Rule WLM106 or Rule WLM107 (one of these rules will be produced depending upon the nature of the service class and performance goal).

  Determine whether most transactions missed the response objective or whether a few transactions **significantly** missed the response objective. If only a few transactions **significantly** missed the response objective, it is likely that these transactions skewed the findings.

- Review your performance goal for the transactions served by the service class, to determine whether the response goal is correct.

- Review the application processing the transactions, to determine whether the application code can more efficiently use the CPU. If the application code can be made more efficient, less CPU time will be required to process the transactions.

  If you find that some transactions skewed the findings, you may wish to consider other alternatives:

  - If you can identify the transactions, perhaps you can use Workload Categorization to place the transactions into a different service class.

You may wish to specify a different importance and different performance goal for this new service class.

- If you do not wish to place the transactions into a different service class (or are unable to identify them), perhaps you can establish another performance period for the existing service class. By specifying an appropriate DUR value, you can cause the SRM to migrate the transactions significantly using the CPU into a lower service class period (perhaps with a different importance and different performance goal).

  This particular alternative is easy to implement, and the inherent processing characteristics of the transactions will automatically cause them to be migrated to lower period service classes. As the CPU-intensive transactions use CPU cycles, they will accumulate service, and the SRM will migrate the CPU-intensive transactions to a lower performance period.

  This alternative is not listed as the initial alternative because the transactions will initially execute in Period 1 of the service class. By executing in Period 1 of the service class, the transactions may deprive short-running transactions of access to a processor and thus cause the short-running transactions to be unreasonably delayed.

- If you have specified an **average response goal** for the service class, perhaps you can change the goal to a **percentile response goal**. With a percentile goal, the Workload Manager would not be as concerned about the few transactions which used significantly more resources and consequently skewed the average response. Rather, the Workload Manager would base its workload management decisions on the percent of transactions which met the response goal.

- If none of the above options are applicable, and if this service class is very important, you may wish to consider running the application on a more powerful processor.

  Note that simply increasing the Importance specified to the Workload Manager, or adding more logical processors (in an LPAR environment) will not resolve the problem with the service class not achieving its response goal. Transactions are delayed because they are using the CPU, not because they are denied access to the CPU[1].

[1]Although other rules also may show that transactions also are denied access to the CPU, Rule WLM200 reports that transactions are delayed because of CPU use.